# Statistical Methods for Data Science

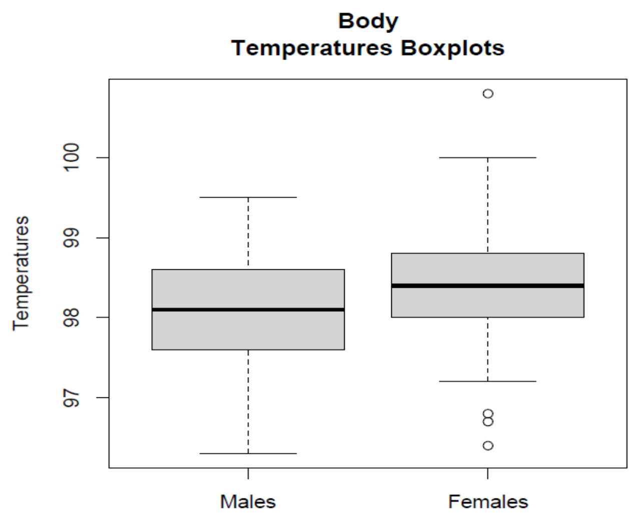
Mini Project #4

Yagna Srinivasa Harsha Annadata

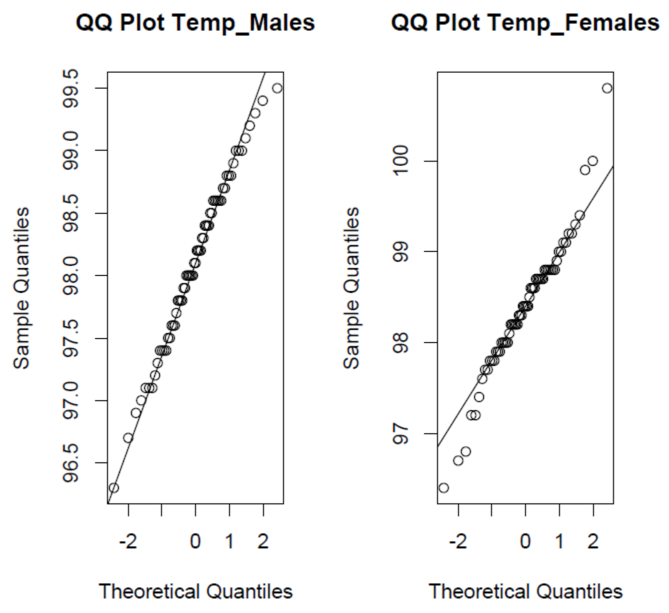
Yxa210024

## Problem1:

### (a):



Females have a slightly higher mean than males and female distribution is volatile than male hence it cannot have same variance.

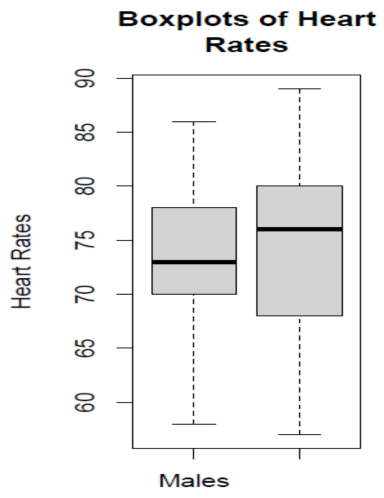


Based on the QQ plot, we can observe that the temperature values for both men and women follow a normal distribution with a similar mean. We can test this hypothesis using a null hypothesis $H0$: means difference = 0, where $m$m and $m$f represent the population mean for males and females, respectively, and an alternative hypothesis $H1$: means difference ≠ 0.

Assuming that the samples are independent and that the distribution is approximately normal, we can use the t-distribution in the Satterthwaite approximation to obtain the confidence intervals. We can construct the confidence interval using the t.test function in R, which gives us a confidence interval of ( -0.53964856, -0.03881298 ) with a p-value of 0.02394.

Since the p-value is less than 0.05 and 0 is not included in the confidence interval, we can reject the null hypothesis and conclude that the mean temperature of females and males is significantly different. The width of the confidence interval is small, indicating that even small deviations from the sample mean can be significant. Moreover, the average female body temperature is slightly higher than that of males.

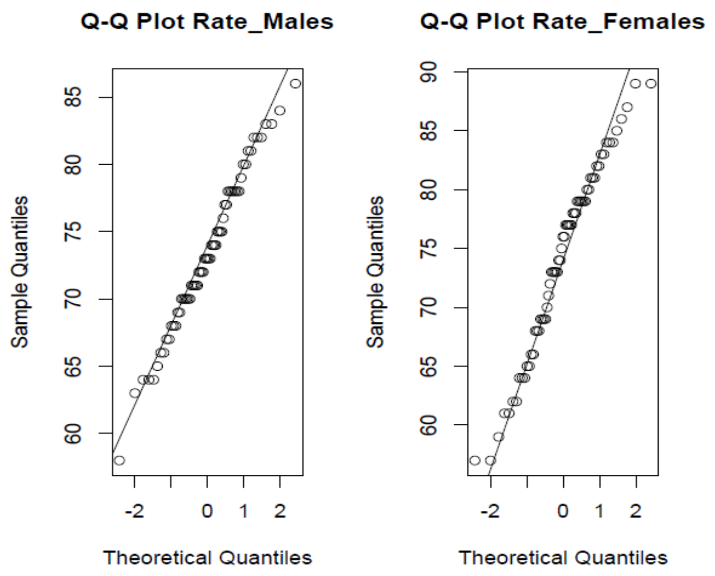
## (b):



The first quartile (Q1) for females is smaller than the Q1 for males, indicating that females have lower values in the lower 25% of the data compared to males. However, the values for females are higher than those for males in the upper 75% of the data, as indicated by the median (Q2) and the third quartile (Q3). This suggests that the distribution of values for females is more spread out, resulting in a higher level of volatility compared to males.

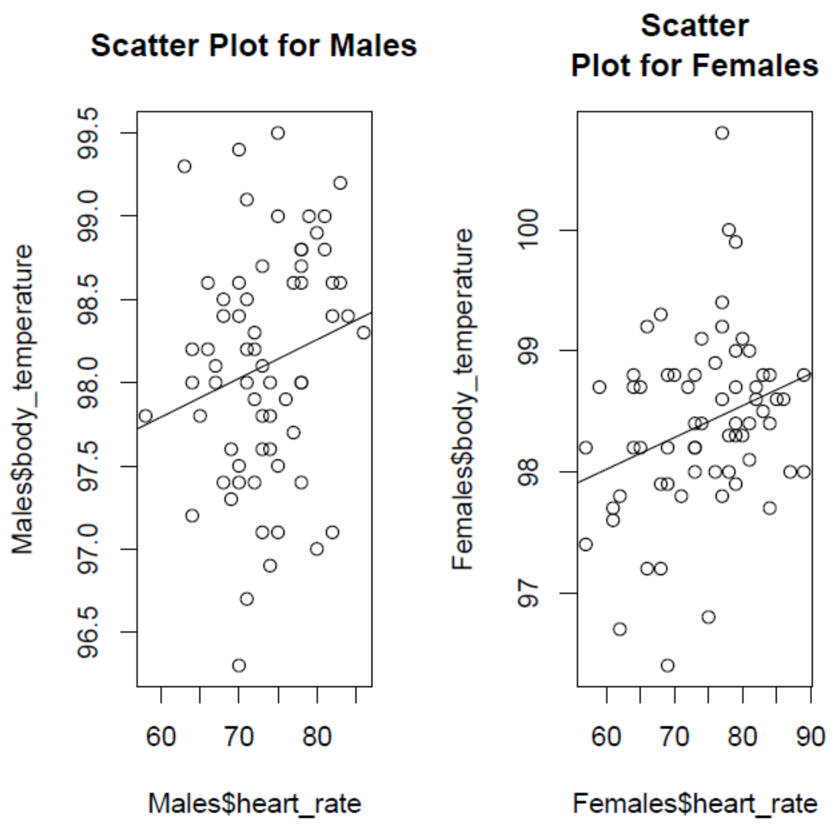
Based on the QQ plot, we can observe that the distribution of heart rate measurements appears to be nearly normal for both men and women. We can test the null hypothesis $H0$: means difference = 0, where $m$m and $m$f represent the population mean for males and females, respectively, and the alternative hypothesis $H1$: means difference ≠ 0.

Assuming that the sample is independent and the variance is nearly equal, we can use the t-distribution with Satterthwaite's approximation to obtain the confidence interval. We can construct the confidence interval using the t.test function in R, which gives us a confidence interval of ( -3.243732, 1.674501 ) with a p-value of 0.5287.



Since the p-value is greater than 0.05 and the value 0 is within the confidence interval, we can accept the null hypothesis and conclude that the mean heart rate is not significantly different between females and males.

(c):



From the graph, it is evident that the slope of the plotted line is greater than 0. This indicates a positive correlation between body temperature and heart rate readings, meaning that as body temperature increases, heart rate tends to increase as well. However, the slope of the line appears to be relatively small, indicating a weak intensity of the linear relationship between body temperature and heart rate.

Based on the provided data, the correlation between body temperature and heart rate is found to be 0.1955894 for males and 0.2869312 for females. Since the correlation values are relatively low, we can conclude that the relationship between body temperature and heart rate is weak.

Additionally, it can be observed that the correlation between body temperature and heart rate is slightly stronger in females compared to males, as the correlation value is higher in females than in males. However, it should be noted that even in females, the correlation is still considered weak based on the given data.

## Rcode:

```
> # Reading data using read.csv function
> Body_temperature_heartrate =
read.csv("C:/Users/yxa210024/Desktop/Masters/spring2023/Stats for
DS/mini_project5/bodytemp-heartrate.csv ", header = T )
> # Separating the two datasets
> Males = subset(Body_temperature_heartrate,
Body_temperature_heartrate$gender == 1)
> Females = subset(Body_temperature_heartrate,
Body_temperature_heartrate$gender == 2)
> # Drawing boxplots for body temperature values
> boxplot(Males$body_temperature, Females$body_temperature, main = "Body
+ Temperatures Boxplots", names = c('Males', 'Females'), ylab =
"Temperatures"
> # Drawing Q-Q plots and confidence interval for the body temperature values
> par(mfrow=c(1,2))
> qqnorm(Males$body_temperature, main = 'QQ Plot Temp_Males')
> qqline(Males$body_temperature)
> qqnorm(Females$body_temperature, main = 'QQ Plot Temp_Females')
> qqline(Females$body_temperature)
> t.test(Males$body_temperature, Females$body_temperature, alternative =
+ 'two.sided', var.equal = F)
Welch Two Sample t-test
data: Males$body_temperature and Females$body_temperature
t = -2.2854, df = 127.51, p-value = 0.02394
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.53964856 -0.03881298
sample estimates:
mean of x mean of y
98.10462 98.39385
> # Drawing boxplots for heart rate values
> boxplot(Males$heart_rate, Females$heart_rate, main = "Boxplots of Heart
+ Rates",names = c('Males', 'Females'), ylab = "Heart Rates")
> # Drawing Q-Q plots and confidence interval for the heart rate values
> par(mfrow=c(1,2))
> qqnorm(Males$heart_rate, main = 'Q-Q Plot Rate_Males')
> qqline(Males$heart_rate)
> qqnorm(Females$heart_rate, main = 'Q-Q Plot Rate_Females')
> qqline(Females$heart_rate)
> t.test(Males$heart_rate, Females$heart_rate, alternative = 'two.sided',
+ var.equal = F)
Welch Two Sample t-test
data: Males$heart_rate and Females$heart_rate
t = -0.63191, df = 116.7, p-value = 0.5287
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-3.243732 1.674501
sample estimates:
mean of x mean of y
73.36923 74.15385
> # Drawing the scatter plots for the body temperature and heart rate values
for males and females
> plot(Females$heart_rate, Females$body_temperature, pch=1, main='Scatter
> par(mfrow=c(1,2))
> plot(Males$heart_rate, Males$body_temperature, pch=1, main='Scatter Plot
for Males')
> abline(lm(Males$body_temperature~Males$heart_rate))
> plot(Females$heart_rate, Females$body_temperature, pch=1, main='Scatter
+ Plot for Females')
> abline(lm(Females$body_temperature~Females$heart_rate)
> # Correlation function
> cor(Males$body_temperature,Males$heart_rate)
[1] 0.1955894
> cor(Females$body_temperature,Females$heart_rate)
[1] 0.2869312
```

## Problem2:

### (a):

Our objective is to generate Monte Carlo approximations of coverage probabilities for two intervals by creating appropriate data sets. We will construct two confidence intervals and repeat the process 5000 times.

Here are the details of the functions we will use:

1. "Zinterval" - It takes input parameters n and $\lambda$ values, simulates a sample, constructs an interval, and determines whether the true mean is within the confidence interval.
2. "zproportion" - It takes input parameters n and $\lambda$ values, calls the "checkzci" function 5000 times, and calculates the coverage probabilities.
3. "Mean_star" - Since we need to generate samples from the distribution, this function returns the mean.
4. "boot_CI" - It takes input parameters n and $\lambda$ values, calls the "Mean_star" function 1000 times, forms a confidence interval, and returns whether the true mean is within the interval.
5. "bproportion" - It takes input parameters n and $\lambda$ values, constructs the parametric initial bootstrap sample, calls the "boot_CI" function 5000 times, and calculates the coverage probabilities.

Using these functions, for the (n,$\lambda$) combination as (5, 0.01) we get the coverage probabilities as:

Z-interval: 0.8074

Bootstrap interval: 0.8976

## (b):

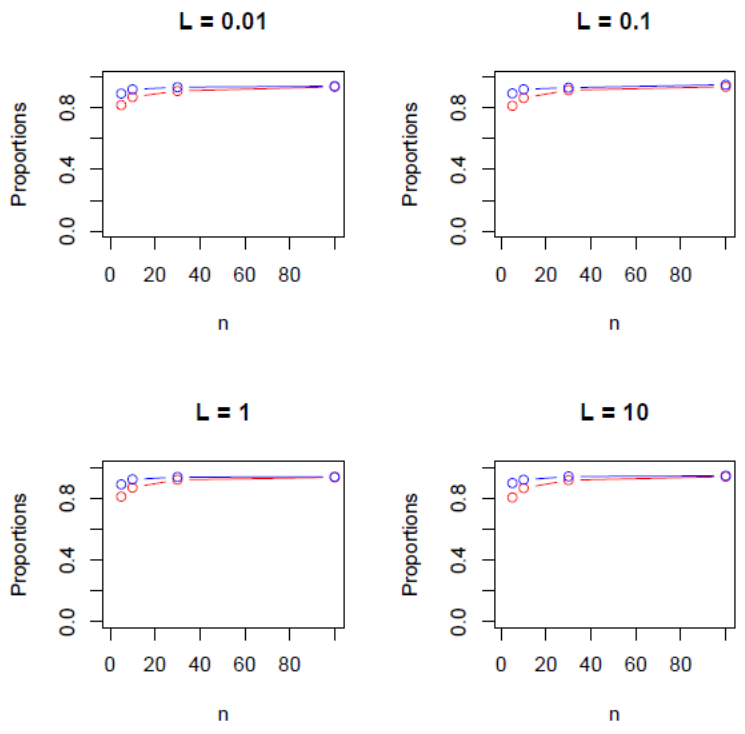
Table1:

| Zproportion | L=0.01 | L=0.1 | L=1 | L=10 |
|---|---|---|---|---|
| N=5 | 0.8182 | 0.8132 | 0.8124 | 0.8064 |
| N=10 | 0.8706 | 0.8640 | 0.8706 | 0.8674 |
| N=30 | 0.9094 | 0.9148 | 0.9212 | 0.9188 |
| N=100 | 0.9360 | 0.9370 | 0.9372 | 0.9414 |

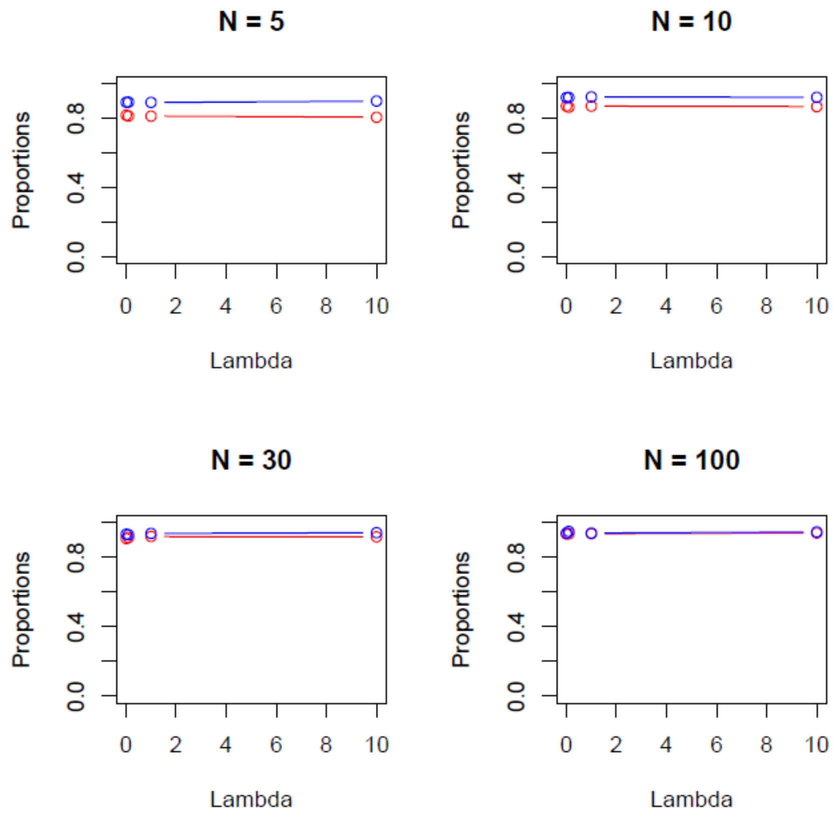
Table2:

| Bproportion | L=0.01 | L=0.1 | L=1 | L=10 |
|---|---|---|---|---|
| N=5 | 0.8924 | 0.8930 | 0.8914 | 0.8998 |
| N=10 | 0.9194 | 0.9204 | 0.9236 | 0.9212 |
| N=30 | 0.9342 | 0.9310 | 0.9390 | 0.9428 |
| N=100 | 0.9408 | 0.9500 | 0.9408 | 0.9476 |

Graph L:



Graph N:



(c):

Based on the analysis of GRAPHS N and L, it can be observed that changing the value of $\lambda$ does not significantly affect the graphs. This suggests that the coverage probabilities are not dependent on $\lambda$. Furthermore, the coverage probabilities obtained via the bootstrap method are higher than those obtained using the z-interval method.

From GRAPHS N, it can be concluded that the coverage probabilities are dependent on the sample size, n. Specifically, for the large-sample z-interval method, the coverage probabilities are accurate when n is large (e.g. n=100). However, for the bootstrap method, the coverage probabilities are consistently higher (approximately) from n=30 onwards.

Considering all the graphs, it can be inferred that the bootstrap method yields higher coverage probabilities for every combination of (n, $\lambda$) compared to the large-sample z-interval method. This indicates that the bootstrap method is more accurate, even for low values of n. Therefore, the bootstrap method is recommended for our problem.

## (d):

### Table 3:

Coverage Probability for bootstrap for lambda = 0.1

| N | Coverage probability |
|---|---|
| 5 | 0.6 |
| 10 | 0.6 |
| 30 | 0.7 |
| 100 | 0.7 |

### Table 4:

Coverage Probability for large sample z for lambda = 0.1

| N | Coverage probability |
|---|---|
| 5 | 0.8 |
| 10 | 0.8 |
| 30 | 0.9 |
| 100 | 0.9 |

## Rcode:

```
# Define function Zinterval
> Zinterval <- function(n, lambda) {
+ A <- rexp(n,lambda)
+ lower <- mean(A) - qnorm(0.975) * sd(A) / sqrt(n)
+ upper <- mean(A) + qnorm(0.975) * sd(A) / sqrt(n)
+ Tmean = 1/lambda
+ if(upper>Tmean & lower<Tmean) {
R Console Page 5
+ return (1)
+ }
+ else {
+ return (0)
+ }
+ }
>
> # Define function zproportion
> zproportion <- function(n, lambda) {
+ CI <- replicate(5000, Zinterval(n, lambda))
+ value <- CI[which (CI == 1)]
+ return (length(value)/5000)
+ }
>
> # Get value of n = 5 and lambda = 0.01 for zproportion
> zproportion(5,0.01)
[1] 0.815
>
```

```
> # Define function Mean_star
> Mean_star<- function(n,lambda) {
+ U_star <- rexp(n, lambda)
+ return (mean(U_star))
+ }
>
> # Define function boot_CI
> boot_CI <- function(n, lambda) {
+ B <- rexp(n,lambda)
+ True_Mean <- 1/lambda
+ lambda1 = 1/mean(B)
+ C <- replicate(1000, Mean_star(n, lambda1))
+ bound <- sort(C)[c(25, 975)]
+ if(bound[2]>True_Mean & bound[1]<True_Mean) {
+ return (1)
+ }
+ else {
+ return (0)
+ }
+ }
>
> # Define function bproportion
> bproportion <- function(n, lambda) {
+ CI <- replicate(5000, boot_CI(n, lambda))
+ value <-CI[which (CI == 1)]
+ return (length(value)/5000)
+ }
>
> # Get value of n = 5 and lambda = 0.01 for bproportion
> bproportion(5,0.01)
[1] 0.8994
>
> # Generate proportion values for bootstrap and z-interval for all
combinations of n and lambda
>
> Z_matrix <- matrix(c(zproportion(5,0.01), zproportion(10,0.01),
zproportion(30,0.01), zproporti
on(100,0.01), zproportion(5,0.1), zproportion(10,0.1), zproportion(30,0.1),
zproportion(100,0.1),
zproportion(5,1), zproportion(10,1), zproportion(30,1), zproportion(100,1),
zproportion(5,10), z
proportion(10,10), zproportion(30,10), zproportion(100,10)),nrow=4,ncol =4)
>
>
> B_matrix <- matrix(c(bproportion(5,0.01), bproportion(10,0.01),
bproportion(30,0.01), bproporti
on(100,0.01), bproportion(5,0.1), bproportion(10,0.1), bproportion(30,0.1),
bproportion(100,0.1),
bproportion(5,1), bproportion(10,1), bproportion(30,1), bproportion(100,1),
bproportion(5,10), b
proportion(10,10), bproportion(30,10), bproportion(100,10)),nrow=4,ncol =4)
>
par(mfrow=c(2,2))
> # Graph for L
> plot(c(5,10,30,100), Z_matrix[,1], main = "L = 0.01", xlab = 'n', ylab
='Proportions', col = 'r
ed',
+ type = 'b', xlim = c(1,100), ylim = c(0,1))
> lines(c(5,10,30,100), B_matrix[,1], col = 'blue', type = 'b')
> plot(c(5,10,30,100), Z_matrix[,2], main = "L = 0.1", xlab = 'n', ylab =
'Proportions', col ='re
d',
+ type = 'b', xlim = c(1,100), ylim = c(0,1))
> lines(c(5,10,30,100), B_matrix[,2], col = 'blue', type = 'b')
>
> plot(c(5,10,30,100), Z_matrix[,3], main = "L = 1", xlab = 'n', ylab =
'Proportions', col = 'red
',
+ type = 'b', xlim = c(1,100), ylim = c(0,1))
> lines(c(5,10,30,100), B_matrix[,3], col = 'blue', type = 'b')
>
> plot(c(5,10,30,100), Z_matrix[,4], main = "L = 10", xlab = 'n', ylab =
'Proportions', col = 're
d',
+ type = 'b', xlim = c(1,100), ylim = c(0,1))
> lines(c(5,10,30,100), B_matrix[,4], col = 'blue', type = 'b')
>
>> # Graph for N
> plot(c(0.01,0.1,1,10), Z_matrix[1,], main = "N = 5", xlab = 'Lambda', ylab
= 'Proportions', col
='red', type = 'b', xlim = c(0.01,10), ylim = c(0,1))
> lines(c(0.01,0.1,1,10), B_matrix[1,], col = 'blue', type = 'b')
>
> plot(c(0.01,0.1,1,10), Z_matrix[2,], main = "N = 10", xlab = 'Lambda', ylab
= 'Proportions', co
l ='red', type = 'b', xlim = c(0.01,10), ylim = c(0,1))
> lines(c(0.01,0.1,1,10), B_matrix[2,], col = 'blue', type = 'b')
> plot(c(0.01,0.1,1,10), Z_matrix[3,], main = "N = 30", xlab = 'Lambda', ylab
= 'Proportions', co
l ='red', type = 'b', xlim = c(0.01,10), ylim = c(0,1))
> lines(c(0.01,0.1,1,10), B_matrix[3,], col = 'blue', type = 'b')
>
> plot(c(0.01,0.1,1,10), Z_matrix[4,], main = "N = 100", xlab = 'Lambda',
ylab ='Proportions', co
l ='red', type = 'b', xlim = c(0.01,10), ylim = c(0,1))
> lines(c(0.01,0.1,1,10), B_matrix[4,], col = 'blue', type = 'b')
>
> # Z Matrix
> Z_matrix
[,1] [,2] [,3] [,4]
[1,] 0.8182 0.8132 0.8124 0.8064
[2,] 0.8706 0.8640 0.8706 0.8674
[3,] 0.9094 0.9148 0.9212 0.9188
[4,] 0.9360 0.9370 0.9372 0.9414
> # B Matrix
> B_matrix
[,1] [,2] [,3] [,4]
[1,] 0.8924 0.8930 0.8914 0.8998
[2,] 0.9194 0.9204 0.9236 0.9212
[3,] 0.9342 0.9310 0.9390 0.9428
[4,] 0.9408 0.9500 0.9408 0.9476
```